Universidad Carlos III de Madrid

Bachelor thesis

Aerospace Engineering

Force decomposition in 3D unsteady aerodynamics

Author:
Carlos Paulete Periañez

Supervisor:
Alejandro Gonzalo Grande

Abstract

Currently, the aerodynamic effects produced in flapping flight are not understood completely. Even though, there are methods to compute the forces acting on three dimensional wings or two dimensional airfoils such as direct numerical simulations, they are to expensive in computational time and fail to give any insight into the actual mechanics of the problem.

In classical (fixed wing) aerodynamics, it is typical to separate the problem into the angle of attack, the camber and the thickness problem. In this way the effect of each design variable for an airfoil is known and design for specific performance is simpler.

In flapping flight, decomposition of the contributions to the force has been implemented in two dimensions, most recently by Martín-Alcántara in [8], by using the algorithm created by Chang in [2].

However, it is known that two dimensional flow does not produce equivalent results to three dimensional flow. Moreover, actual designs of flapping vehicles will need to take three dimensional effects into account to yield a high performance.

In this paper, a tool to compute the force decomposition described by Chang [2] is developed. This methodology requires the calculation of a surface potential. So the library BEMLIB [10], was modified to solve the potential flow in two and three dimensions over wings and airfoils. The library was validated against analytical results in the two dimensional case and known numerical results in three dimensions.

For validation purposes, the decomposition algorithm was applied to a wing describing sinusoidal heaving motion. The results of this decomposition are presented and the limitation and requirements of the decomposition algorithm discussed.

Contents

1 Introduction

1.1 Motivation

Throughout the last half of the twentieth century and into the present, vast knowledge in the field of unsteady aerodynamics has been accumulated. Particularly, the flapping flights of living creatures such as insects [3]. However, the progress in the design of flapping micro air vehicles (FMAV) has been stale. This is in the end the objective of this work, to advance the understanding of flapping flight in order to facilitate future development of FMAVs.

There are, most likely, three main reasons for this slow down.

Most of the research up to the 2000s focuses on observing living specimens and trying to relate movements observed with forces generated, this may give an idea of how a certain organism flies but it fails in the prediction of the forces produced by other systems.

Numerical analysis has been used to a great extent in the field, mostly in recent years with current advances in computing technology. These methods, although accurate are extremely expensive in time and resources, which couples with the last problem.

Associated with flapping flight, a huge design space is found. Whereas to design a quad rotor drone or a model aircraft basically one needs a relatively simple geometry. To design a FMAV, one needs not only to design the geometry but also the cyclic motion associated to that geometry that generates an efficient flight profile.

This last thing mentioned, the motion of the wings, is particularly relevant. Designing a FMAV means designing an insect size flying machine, which must be able to react to its environment, be it environmental conditions or objects, and it must do it very quickly. Being the size of the vehicle so small, it greatly restricts the computing capacity it is capable of carrying. To create this control system, a reduced system is required such that a low power computer can anticipate the effects of a wing movement and use it to correct a trajectory in real time.

As in classical aerodynamics, where is typical to split the contributions to a force into several contributions, in this paper, the force decomposition developed by Chang [2] was implemented so that a model of all or several of the contributions to the forces of a flapping vehicle can be built and optimized.

1.2 State of the art

The aerodynamic effects that appear in the flapping flight of living creatures, such as insects or birds, are very complicated in its nature, a recent description of this problem and its mechanics can be found in [13]. For this reason, its development is in a much earlier stage than the case of fixed wings, for instance. The flapping flight problem, can be simulated and has been done successfully using direct numerical simulation (DNS) techniques [8] [9]. However, as it is widely known, although complex models (such as DNS) give accurate results, simple models give insight into the fundamental effects acting on the system.

Trying to simplify the problem, the scientific community, has tried to reduce the problem from three dimensional (3D) wings to two dimensional (2D) airfoils, that represent infinite aspect ratio (AR) wings. Many studies have been published studying the aerodynamic characteristic of heaving airfoils [15] [7].

Nevertheless, creatures that make use of this kind of flight, not only beat their wings up and down in heaving but pitch them cyclically to maximize efficiency. To try to model this, some authors have included pitching into their studies of airfoils [1].

As in classical aerodynamics, to try to better understand the characteristics of the motion, some authors have implemented decomposition algorithms such as that created by Chang [2] and used in this paper. In these papers, 2D cases have been decomposed into simpler constituents [8] [14].

Just in 2D, the size of the design space up to this point has increased greatly compared with fixed wing flight. Not only must the shape of the wing be chosen but also the pitching motion and the heaving motion. Then, going into 3D models, it is even larger including the design of the flapping motion with parameters such as the radius of rotation of the wing and the amplitude.

Currently, some researchers are trying to get insight out of 3D flapping and heaving wings using all methods described above. The task is much harder and complex than for 2D airfoils, but the fact is that, flows in 3D are the reality of birds and insects. Most of them present low aspect ratios in their wings accentuating the errors committed assuming infinite span in 2D. From this line of research, this paper is created to facilitate insight and simplify the problem.

1.3 Objectives

The main goal of this work is to implement a tool capable of decomposing the aerodynamic forces produced by flapping wings following the formulation proposed

by Chang [2]. This formulation splits the aerodynamic forces in three contributions (namely, surface vorticity (SV), vorticity within the flow (VwiF) and added mass (AM)), which depend of a potential flow and of fluid elements of non-zero vorticity.

In order to calculate the potential flow of the problem under study, part of the boundary element method library developed by Pozrikidis (BEMLIB) [10] has been modified.

Since the formulation of Chang [2] also considers the vorticity on the wing surface and in the flow around the wing to decompose the aerodynamic forces, 3D flow data are required by the tool to obtain those forces. These data are extracted from a direct numerical simulation of the problem studied during this work.

2 Methodology

In this section, the problem to be solved is explained as well as the numerical method used to solve it. It will be divided into 5 subsections each one required to obtain the final tool developed in the context of this work

Firstly, in section 2.1, the force decomposition algorithm devised by Chang [2], is derived from the Navier-Stokes equations to the final integral form used later on.

Then, in section 2.2, the parts of the BEMLIB library used in the resolution of the auxiliary potentials is presented and the method used in them to solve the potential explained.

The next two sections 2.3 and 2.4 cover the changes made to the 2D and 3D libraries to optimize and increase their capabilities, as well as the validation procedures followed when the changes where implemented.

Lastly in section 2.5, the decomposition algorithm itself is broken down and discussed. And the different program routines required to break down the forces into its constituents, listed and detailed.

2.1 Force decomposition

The total force acting on the wing is divided according to the algorithm proposed by Chang in [2]. Here, a full derivation of the force decomposition algorithm is presented. This derivation is extracted from Moriche [9] and is shown in this paper for completeness.

The algorithm uses the potential ϕ_i, where the subindex i stands for each of the potential flow directions used to calculate the potential. This potential is defined by the following boundary conditions:

$$\begin{aligned} \nabla^2\phi_i &= 0 && && \\ \nabla\phi_i \cdot \vec{n} &= -\vec{n} \cdot \vec{e}_i U_\infty && (1) && \text{At the body surface} \\ \phi_i &\to 0 && && \text{At infinity} \end{aligned}$$

Shown here for the x-direction, where $\vec{n}$ defines a vector normal to the surface and pointing toward the fluid and U_∞ is the free stream velocity.

From the Navier-Stokes momentum equation:

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\vec{u} = -\nabla p + \nu \nabla^2 \vec{u}, \tag{2}$$

this equation can be manipulated to remove the pressure term which is known to generate errors in incompressible fluid numerical simulations [12]:

$$\begin{aligned} -\int_V \frac{\nabla \phi_i}{U_\infty} \cdot \nabla p \; dV = &-\int_V \frac{\nabla \phi_i}{U_\infty} \frac{\partial \vec{u}}{\partial t} \; dV + \frac{1}{2} \int_V \frac{\nabla \phi_i}{U_\infty} \cdot \nabla(|\vec{u}|^2) \; dV \\ &- \int_V \frac{\nabla \phi_x}{U_\infty} \cdot (\vec{u} \times \vec{\omega}) \; dV + \nu \int_V \frac{\nabla \phi_i}{U_\infty} \cdot (\nabla \times \vec{\omega}) \; dV, \end{aligned} \tag{3}$$

note that, the previous equation (3), is determined by volume integrals. However, is more desirable to rewrite them in terms of surface integral to speed up numerical calculations. To that end, the divergence theorem ($\int_V \nabla \cdot \vec{F} \; dV = \int_S \vec{n} \cdot \vec{F} \; dA$) can be used in addition to the following vector calculus identities.

$$\nabla \cdot (f\vec{A}) = f \nabla \cdot \vec{A} + \vec{A} \cdot f \tag{4}$$

$$\nabla \cdot (A \vec{\times} \vec{B}) = \vec{B} \cdot \nabla \times \vec{A} - \vec{A} \cdot \nabla \times \vec{B} \tag{5}$$

The term on the left hand side can be successfully transformed by using the identity (4), yielding:

$$\frac{\nabla \phi_i}{U_\infty} \cdot \nabla p = \nabla \cdot (p \frac{\nabla \phi_i}{U_\infty}) - \cancelto{0}{p \nabla \cdot \frac{\nabla \phi_i}{U_\infty}} = \nabla \cdot (p \frac{\nabla \phi_i}{U_\infty}), \tag{6}$$

now the first term on the right hand side is treated using the same identity (4) to obtain:

$$\frac{\nabla \phi_i}{U_\infty} \frac{\partial \vec{u}}{\partial t} = \nabla \cdot (\frac{\phi_i}{U_\infty} \frac{\partial \vec{u}}{\partial t}) - \cancelto{0}{|\vec{u}|^2 \nabla \cdot \frac{\nabla \phi_i}{U_\infty}} = \nabla \cdot (\frac{\phi_i}{U_\infty} \frac{\partial \vec{u}}{\partial t}) \tag{7}$$

The next term on the right hand side can also be rearranged using identity (4):

$$\frac{\nabla \phi_i}{U_\infty} \cdot \nabla(|\vec{u}|^2) = \nabla \cdot (|\vec{u}|^2 \frac{\nabla \phi_i}{U_\infty}) - \cancelto{0}{|\vec{u}|^2 \nabla \cdot \frac{\nabla \phi_i}{U_\infty}} = \nabla \cdot (|\vec{u}|^2 \frac{\nabla \phi_i}{U_\infty}) \tag{8}$$

Using (5) the last term on the right hand side can be expressed as:

$$\frac{\nabla\phi_i}{U_\infty}\cdot(\nabla\times\vec{\omega}) = \nabla\cdot(\vec{\omega}\times\frac{\nabla\phi_i}{U_\infty}) - \vec{\omega}\cdot\cancelto{0}{\nabla\times\frac{\nabla\phi_i}{U_\infty}} = \nabla\cdot(\vec{\omega}\times\frac{\nabla\phi_i}{U_\infty}) \tag{9}$$

Subsequently, using the divergence theorem one can express every term as surface integrals, excluding the third term on the right hand side :

$$\begin{aligned}\int_S p\frac{\nabla\phi_i}{U_\infty}\cdot\vec{n}\ dA = &-\int_S\frac{\phi_i}{U_\infty}\frac{\partial\vec{u}}{\partial t}\cdot\vec{n}\ dA + \frac{1}{2}\int_S|\vec{u}|^2\frac{\nabla\phi_i}{U_\infty}\cdot\vec{n}\ dA\\ &-\int_V\frac{\nabla\phi_i}{U_\infty}\cdot(\vec{u}\times\vec{\omega})\ dV - \nu\int_S\vec{n}\times\vec{\omega}\cdot\frac{\nabla\phi_i}{U_\infty}\ dA\end{aligned} \tag{10}$$

From the boundary conditions of the problem (1) stated above, equation (10) can be expressed as the pressure force acting on the wing as:

$$\int_S p\frac{\nabla\phi_i}{U_\infty}\cdot\vec{n}\ dA = \int_S p\vec{n}\cdot\vec{e}_i\ dA \tag{11}$$

Therefore, if contribution to the force due to the viscous shear is added, an expression for the total force can be obtained:

$$\begin{aligned}F_i &= -\int_S p\vec{n}\cdot\vec{e}_i\ dA + \nu\int_S(\vec{\omega}\times\vec{n})\cdot\vec{e}_i\ dA\\ &= -\int_S\frac{\phi_i}{U_\infty}\frac{\partial\vec{u}}{\partial t}\cdot\vec{n}\ dA + \frac{1}{2}\int_S|\vec{u}|^2\frac{\nabla\phi_i}{U_\infty}\cdot\vec{n}\ dA\\ &\quad -\int_V\frac{\nabla\phi_i}{U_\infty}\cdot(\vec{u}\times\vec{\omega})\ dV - \nu\int_S(\vec{\omega}\times\vec{n})\cdot(\frac{\nabla\phi_i}{U_\infty}+\vec{e}_i)\ dA\end{aligned} \tag{12}$$

With equation (12), the expression for the force acting on the wing instead of being expressed in pressure and viscous terms, it is decomposed in four terms that do not involve the pressure. This ensures that no problems will appear due to the peculiar behavior of the pressure in incompressible flows [12].

The first two terms on the right hand side represent the added mass contribution, which takes into account the inertia added to the system due to the acceleration of the wing and the consequent motion of the fluid volume around it. Accordingly, as

will be shown later, this term is more relevant when the wing moves with maximum acceleration.

The third term is a result of the vorticity of the flow in the control volume and finally, the last term is a consequence of the surface vorticity. This is completed with the definition of the potential, which depends only on the shape and orientation of the wing.

Equation (12) shows the relevance of ϕ in the computation of the force, this calculation can be done using a boundary element method, in this work, the library BEMLIB [10] is modified to is modified to solve the potential flow around complex geometries. A description of the library as well as the modifications applied to it are explained in the next section.

2.2 The BEMLIB library

The BEMLIB library [10] are a set of Fortran 77 and Matlab files that solve for Laplace's equation applying boundary element methods. The library also includes files to discretize limited two and three dimensional geometries.

Boundary element methods are used to solve partial differential equations such as Laplace's equation and Helmholtz's equation. The main advantage of these methods, is that they do not require solving for the objective function in the whole domain, only in the required boundary. In the case studied, for instance, the potential flow over a wing, can be solved using exclusively the wing where a no penetration boundary condition is imposed.

To solve numerically for these equations, BEMLIB discretizes the boundary geometries. In 2D, the boundary line, is given by points joined by straight or curved lines. For a variety of geometries. In 3D, triangular discretization is used, where the surface of the body is defined by a mesh of triangles, in this case unstructured. For the 3D case, BEMLIB only provides the mesh of spheroids that are a result of scaling a sphere in one or several axes.

Although not defined intrinsically in BEMLIB, it is important to define the frame of reference used in this work. Outside of circles used to validate in 2D, only NACA 0012 airfoils and wings made of those airfoils are used. However, the reference frame is not that one typically used in aerodynamics where x points in the direction of travel, z down in the direction of gravity and y points to the right side of the wing. For simplicity, a reference frame that shares two of its components in 2D and 3D was used. Figure 1, shows an airfoil, which would be as well a cut of the wing, with the reference frame superposed.

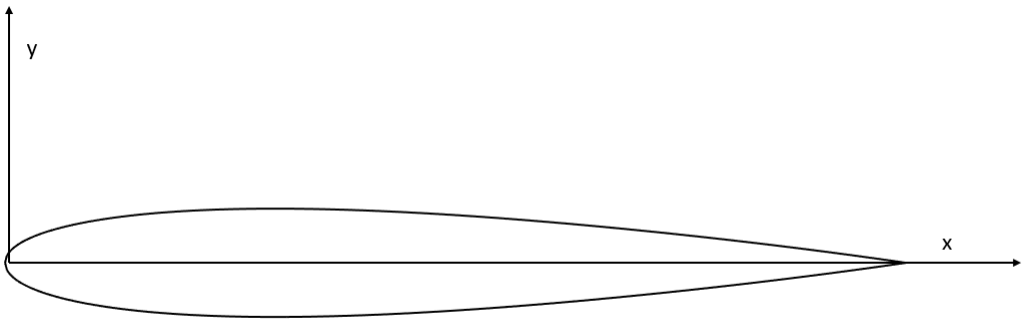


Figure 1: NACA 0012 airfoil with its corresponding reference frame

As seen in figure 1, the reference frame is centered at the leading edge and in 3D wings at the midspan. Axis x points in the flow direction, while axis y points along the vertical pointing up. The z axis in 3D wings, is perpendicular to the other two axis and points toward the left side of the wing.

From the stock library, 2 subcategories are mostly relevant to this work: BODY_2D and LNM_3D.

As their name suggests, BODY_2D is used to solve the potential in 2D and LNM_3D to solve in 3D. They both solve Laplace's equation in each domain respectively, with a submerged body. The workings of each of the branches, the modifications performed on them and the following validation are described on following sections 2.3 and 2.4.

2.3 2D potential

This section involves the validation and changes made to the 2D part of the software library that calculates the potential over a 2D body using boundary element methods.

The potential in 2 dimensions is solved for using the BODY_2D branch of BEMLIB. It solves the two dimensional potential flow past a submerged body applying on its surface non-penetration Neumann boundary conditions. The current implementation can solve for the surface potential in both x and y directions. However, the limitations imposed on the input body shape require a new patch that increases this capacity. For this a Matlab code has been developed in this work that interfaces with the BEMLIB library and can generate shapes such as NACA 4 digit airfoils.

The two dimensional library was only worked on for validation purposes. The real interest lies on solving for the potential in 3D shapes.

Firstly, a validation of the stock library was performed, then the changes made to the library are discussed as well as auxiliary programs developed. Finally the modified program is validated.

Base 2D library validation

The original library (i.e. without any modifications to the source code) allows to solve the potential flow around a circle of a given radius to be given as input. Since this problem has analytic solution, the code can be validated comparing both solutions. The analytic expression given for an ellipse at a certain angle of attack (Remember that a circle is just an ellipse with major axis equal to its minor axis) is extracted from Martín-Alcántara [8]. These expressions are as follows:

$$\phi_x = \frac{cU_\infty}{2}\sqrt{\frac{1+\epsilon}{1-\epsilon}}e^{-\xi}(\sin\theta\sin\eta + \epsilon\cos\theta\cos\eta) \tag{13}$$

$$\phi_y = \frac{cU_\infty}{2}\sqrt{\frac{1+\epsilon}{1-\epsilon}}e^{-\xi}(\cos\theta\sin\eta - \epsilon\sin\theta\cos\eta) \tag{14}$$

Where c is the major axis of the ellipse, ϵ is the ratio of the major and minor axis, θ is the angle of attack of the ellipse and ξ and η are the radial and angular elliptic coordinates. In order to introduce a circle, in equations (13) and (14), ϵ must be equal to one. However, this yields a zero in the denominator. This can be solved by introducing ϵ very close to 1.

Figure 3 and 4, show the results of these analysis. On each of them, the result from the 2D branch of BEMLIB is plotted with the analytic solution. In the potential distribution, the peaks and valleys, are expected at the regions whose normal to the incoming flow, this is where the intensity of the sources in the distribution must be higher to apply the no penetration boundary condition. In between those regions, a smooth transition is expected.

The angle α used on figures 3 and 4 is sketched in figure 2.

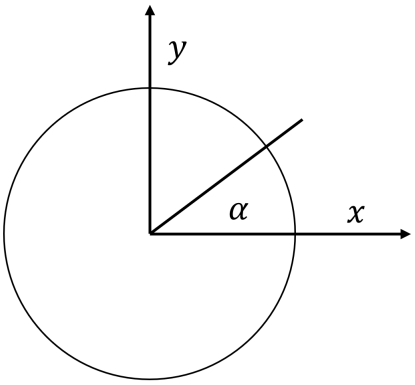


Figure 2: Angle α used for representation in figures 3 and 4

In figure 2, the angle α sketched is the angle used in figures 3 and 4, the x and y axes are both directions of the potential flow used to calculate the boundary potential distribution.

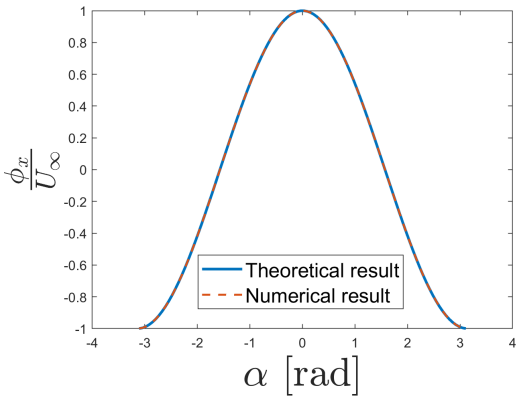


Figure 3: Analytic and numeric ϕ_x

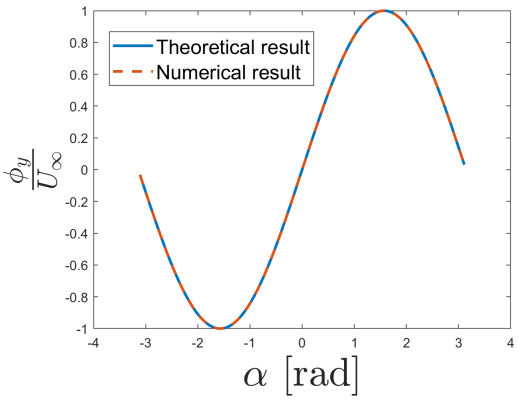


Figure 4: Analytic and numeric ϕ_y

As expected, the peaks and valleys are produced at the regions where the flow is directly incident. That is to say, with the potential flow in the x direction, the maximum and minimum happen at $\alpha = 0rad$ and $\alpha = \pi rad$. And when the flow is the y direction, due to the cylindrical symmetry of the problem, the maximum and minimum happen at $\alpha = -\frac{\pi}{2}rad$ and $\alpha = \frac{\pi}{2}rad$.

Looking at the similitudes between the analytic and BEMLIB potential, an almost perfect match is produced throughout the whole boundary. This result is very important since it means that the library works correctly without any modifications

made to it. Thus, further work can be extracted from it and modifications can be applied to expand its capabilities.

Point cloud input validation

After validation of BODY_2D, a modification was made in order to be able to input a general geometry to it.

To do this, a matlab script was made to interface with BEMLIB and provide a 4 digit NACA profile. The results from this profile were later compared with results for a potential flow through a NACA 0012 calculated by Moriche [9] with other boundary potential tool.

The resulting potential from both methods is shown in figures 5 and 6, using the reference frame defined in 1.

Figure 5, shows the potential distribution for a x direction flow over the foil. As in section 2.3, the minimum in the potential, is expected where the normal to the surface has equal direction but opposite direction to the incoming flow, that is, the leading edge. However, since the rear section, i.e. the trailing edge, is much sharper than in a circle such as the one used for validation in 2.3. Thus the maximum found in figure 3 is not expected to be reached in figure 5.

Since the airfoil used is symmetric, the projection of the normals in the x direction is equal for the upper and lower side, then, it follows that the potential distribution is equal on both sides. Because of this, a single line appears on figure 5, although both sides are plotted.

The opposite effects happen in figure 6, where the potential flow now impinges the wing on the lower region along the y axis.

While in figure 5, a low potential distribution (small distance from maximum to minimum) was observed, because most surface normals are aligned perpendicular to the flow, in figure 6, they are aligned with the flow. Thus a large difference is found between the minimum and maximum potential.

In figure 6, a difference between the upper and lower region is observed contrary to the distribution for potential flow in the x direction, this appears because the projections of the normals on the y axis on the upper and lower regions are opposite to each other.

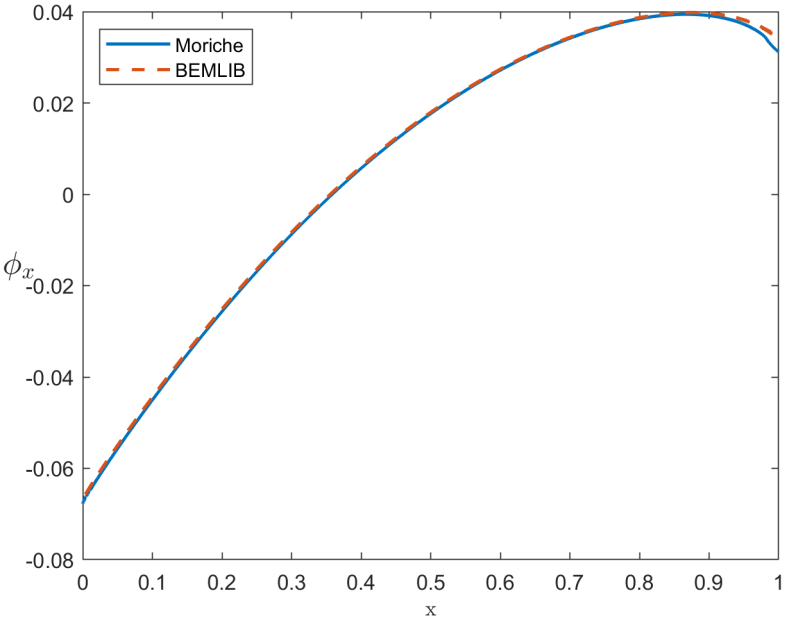


Figure 5: Distribution of potential ϕ_x

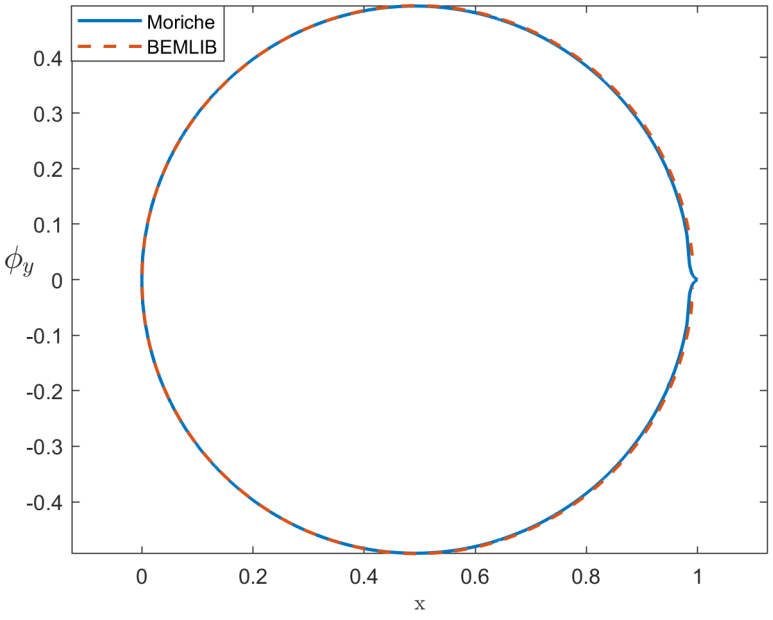


Figure 6: Distribution of potential ϕ_y

Looking at the differences between the solution from Moriche [9] and the solution from BEMLIB, from the leading edge to the beginning of the trailing edge ($x = 0.9c$ where c is the chord) there is almost a perfect match. However, some differences can be observed in the trailing edge region.

The differences seen can be explained by the definition of the foil shape used in each method.

Whereas the foil defined by Moriche [9] his paper has a sharp trailing edge, the foil in this work has a truncated trailing edge in order to preserve the normal vectors in this region. For this reason, the slight point seen at $x \approx 1$ in figure 6 for Moriche's solution is not seen using the modified BEMLIB library.

It is important to remark why the ϕ_y potential shown in 6 is almost symmetric in the chordwise direction. The y-direction potential takes the component of the normal vectors in the y-direction. The projection of the normals at each face on to the y axis ($\vec{n}\vec{e_y}$) can be seen in figure 7.

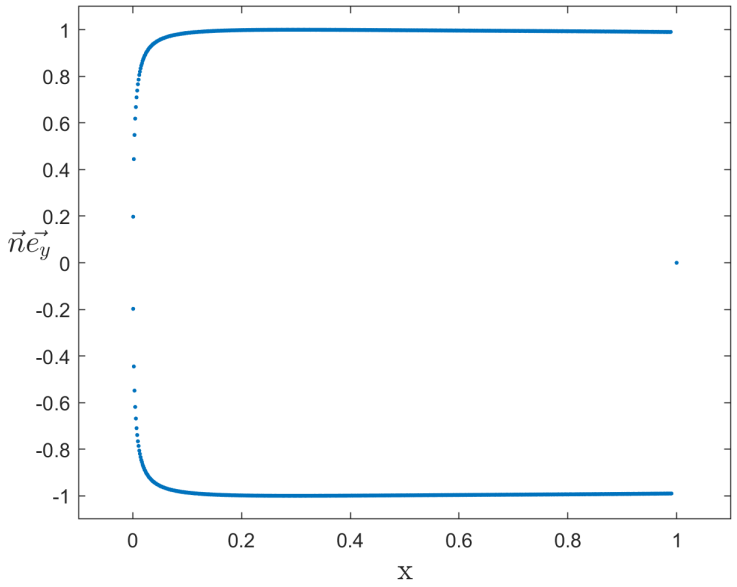


Figure 7: Projection of the normal vector onto the y axis.

From figure 7, it becomes clear that the y component of the normal vector is practically constant from $x = 0.15$ to $x \approx 1$. Then, the only meaningful difference between the leading and trailing edge is the rate with which the projection of the normal changes. While in the leading edge there is a smooth transition from the

upper to the lower side, on the trailing edge there is a sharp transition with only one point with a projection equal to zero since it is in the trailing edge and points in the x direction.

With these results, the 2D code can be considered validated and the focus can be turned to validating the 3D case.

2.4 3D potential

In order to calculate the force decomposition, the potential in 3D must be solved for any desired geometry. In this section, the changes to the 3D branch of BEMLIB are listed and explained. Then, the finished library is validated against the 2D solution found in section 2.3.

This branch, as BODY_2D, solves the Laplace equation with Neumann boundary conditions to obtain the potential at the surface nodes. It does so solving the integral that appears applying Green's third identity at the boundary. The mesh used to discretize the wing is composed of unstructured triangles. For further details on the methods used by BEMLIB one can refer to [10].

The input shapes are severely limited to only spheroids of low to medium resolution which is not nearly enough for the scope of this work. The requirements were set such that any geometry may be inputted into the program. To generate this geometries, Gmsh was used [5].

Gmsh is an open source meshing tool specialized in creating unstructured meshes. Critically, the input to Gmsh is given in ASCII files using its own scripting language. In this work, this is ideal because a Matlab routine can be created to generate these input files given a few parameters of the wing such as the aspect ratio or the maximum mesh element size. Then, the output from Gmsh can be converted from its own file type into a format legible by BEMLIB. Thus a new mesh can be defined, created and given as input to BEMLIB at runtime.

Furthermore, the way the library handled the boundary conditions was unable to accept the ones required to calculate the potential. For this additional modifications to the core code were conducted.

The full list of modifications are discussed in detail in following lines.

LNM_3D modifications

The standard LNM_3D library received a battery of changes listed here:

- **Removed keyboard input requirement**: The program initially needed user input during runtime to finish. This was removed and automated in the Matlab program discussed in section 2.5.
- **Added boundary condition**: The necessary non-penetration Neumann boundary condition $\nabla\phi_i = -U_\infty \vec{n}\vec{e_i}$ was implemented for each of the cartesian directions.
- **General geometry input data**: Paired with a matching Matlab routine, this allows LNM_3D to receive a general Gmsh mesh as geometry input. Allowing for example wings to be solved.
- **Improved linear system solver**: The standard library was prepared to compute the potential flow in one direction. Here, three are required, one per Cartesian direction. This means that the original problem to be solved having one influence matrix and one right hand side, now has one influence matrix and three right hand sides. It was decided that the best course of action was to completely replace the solver for a new one using LU decomposition based on the algorithm in [11]. This new linear system solver is able to obtain the solution of the potential flow in the three directions in less time than that required for the previous solver to get the solution in only one direction.
- **Removed control volume potential computing routine**: LNM_3D, as standard, has the capacity to compute the potential at any point in space. This section was removed and the functionality was transferred to part of the main Matlab program. This saves time because in most cases the potential only has to be computed once (because geometry and orientation do not change) but the potential in the CV must be computed several times (because the wing is at different locations during a cycle). In this way the calculation of the surface potential and of the CV potential can be separated.

This changes made the BEMLIB library capable of fitting inside the main Matlab routine discussed in section 2.5.

Modified LNM_3D validation

Before using the modified library, it is important to compare it against known results to validate it. In this case, this is done comparing the known solution of the potential over a NACA 0012 airfoil in the x and y direction against a cut at the midspan of a wing made of the same kinds of airfoils. The results should match exactly for an infinite span 3D wing but for the sake of calculation time, an $AR = 8$ wing was used.

In figures 8 and 9, in order to compare a 2D solution (from section 2.3) and a 3D solution, the potential flow over the finite $AR = 8$ wing was sliced at the midspan. That is to say that the solution at the neighboring points to the midspan plane were interpolated onto it. Yielding the closest approximation to the 2D potential flow from the finite wing distribution.

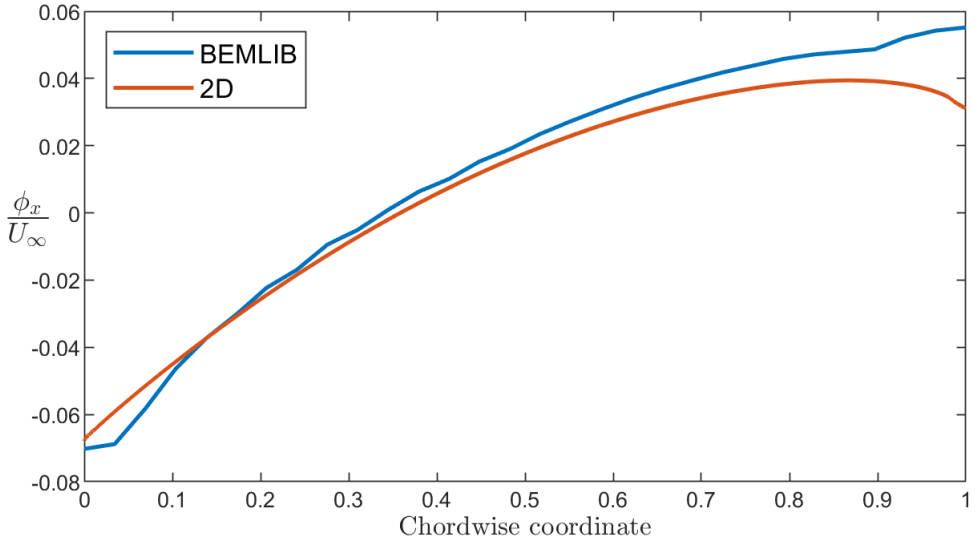


Figure 8: Normalized ϕ_x potential along a NACA 0012 airfoil

In figure 8, the solution the 2D potential in the x direction from [9] is compared against the one computed with the modified 3D branch of BEMLIB. There is a good match between the two all along the chord, diverging slightly at the trailing edge. This phenomenon has been investigated at different xy cuts along the span of the wing. The difference observed at that region between both solutions increases as the distance to the midspan increases. So one can assume that this difference is associated to the finite aspect ratio of the wing in BEMLIB.

Similar results were obtained for the potential flow in the y direction. Figure 9, only shows the result on the upper side of the wing. As in figure 6, remember that the symmetric nature of the foil used, the potential flow in the y direction is also symmetric.

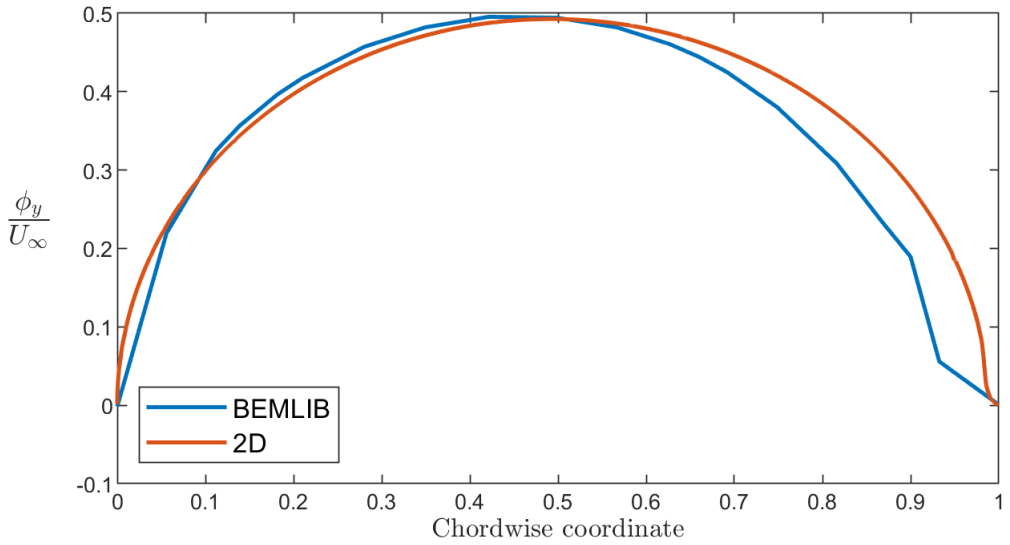


Figure 9: Normalized ϕ_y potential along a NACA 0012 airfoil

In figure 9, there is a good match between the 2 lines up until the trailing edge section is reached. Again, this can be associated to the finite span of the wing.

Moreover, the calculation needed for the 3D potential is much more expensive than that of the 2D potential. This in turn restricts the mesh definition achievable. That is the reason behind the jaggedness of the BEMLIB plot. This can be seen in figure 10, where the mesh of the wing is shown with the potential flow in the x direction overlaid.

While the mesh is fine enough, (as will be seen in section 2.5.2), the number of points along any single xy plane cut, is much smaller than the number of points used by Martín-Alcántara in [8] yielding much more jagged results. A top-down view of the mesh used in this validation can be seen in figure 10.

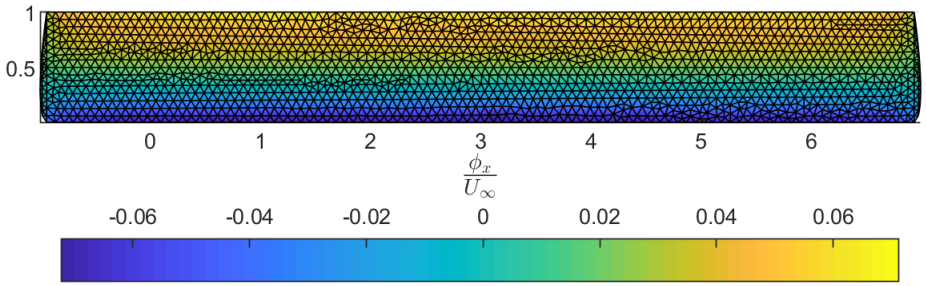


Figure 10: AR 8 wing mesh with ϕ_x color overlay

Seeing the results from figure 8 and 9, one can consider the BEMLIB library validated, and suitable for further calculations.

2.5 3D force decomposition

In order to decompose the forces acting on a wing describing a general flapping/heaving motion following the algorithm proposed by Chang [2], a Matlab program that wraps all the calculations has been developed during this thesis.

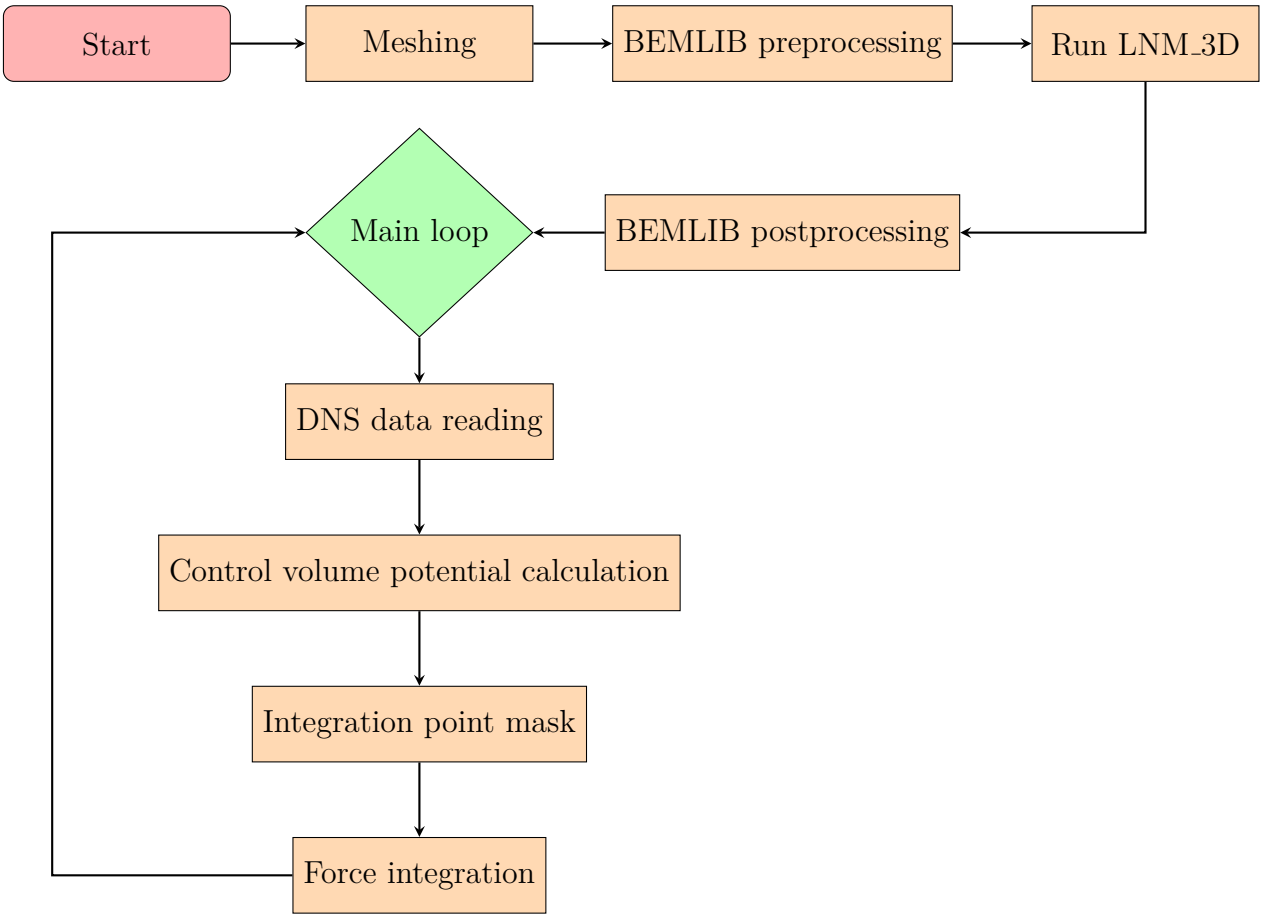
2.5.1 Description of the force decomposition algorithm implemented in Matlab

The Matlab program created in the context of this work, requires no user input besides the initial configuration and calculates the contributions of the force in the three spatial directions for one period of motion. The main program can be broken down into several parts that will run sequentially within the a loop that cycles through each position of the wing during a motion cycle.

While in a general flapping/heaving case, all parts of the program must be within the loop[1], in the heaving case studied here, since the surface potential on the wing is only affected by the geometry itself and the angle at which it is presented to the flow, one can calculate the surface potential flow of a single wing in heaving once and use it throughout the motion. Thus acclerating computations.

[1]The loop here, refers to the steps through a cycle of motion.

Here, a flowchart describing the program structure for the case of a heaving wing is shown.



Each part (shown as an orange rectangle in the flowchart above) will be discussed in detail in the following pages.

Meshing

In order to discretize the wing surface, a surface mesh is required. The mesh itself is created using the free meshing software Gmsh [5].

Gmsh is particularly useful in this scenario because it provides the necessary triangular unstructured mesh required by LNM_3D with a minimal number of input parameters (besides the geometry definition itself). The only parameters given to

Gmsh through the Matlab wrapper are the meshing algorithm and the maximum element size. These parameters are written to the Gmsh configuration file with the description of the geometry (in this case a wing with NACA0012 cross-section and rounded tips as the one shown in figure 11).

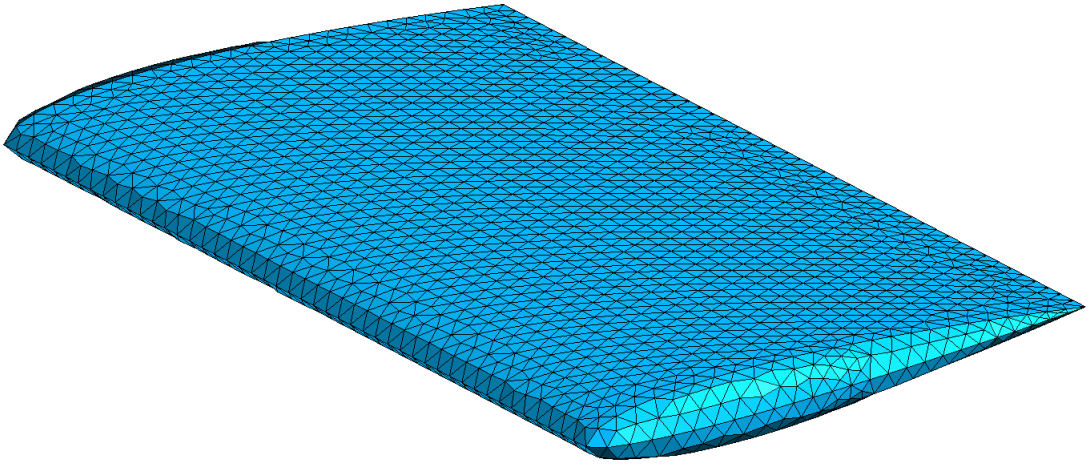


Figure 11: Unstructured triangular mesh of a wing

The meshing algorithm chosen is called frontal in the Gmsh documentation. This algorithm was chosen because although it is less robust (Fails more often than traditional meshing algorithms such as Delaunay) and requires more computational resources, it creates triangles closer to equilateral simplifying the surface integration further on in the program. Since the mesh is quite small, the efficiency of the algorithm is not very important. Gmsh takes less than one minute to generate the mesh in the densest case studied.

Since the meshing algorithm is defined to frontal by default, the only mandatory input parameter to the Matlab program was the maximum element size, called 'CharacteristicLenghtMax' in the Gmsh documentation [5]. This parameter has a major importance in the mesh which must be fine enough that reliable results are obtained but no more, as this would increase calculation time undesirably. For this reason a mesh study is conducted in section 2.5.2 to determine the optimal element size.

LNM_3D pre and post processing

LNM_3D requires a very specific set of variables to calculate the surface potentials correctly. This input is created with Matlab using the mesh previously generated with Gmsh. Concretely, the coordinates of the vertices of each triangle and their connectivity are first read in Matlab from the output file of Gmsh.

Ten the auxiliary points used by Gmsh to create the geometry that do not belong to the mesh itself, must be deleted along with their corresponding indexes in the connectivity matrix[2].

Since the branch LNM_3D requires that the triangular elements are defined by 6 points (as shown in figure 12).

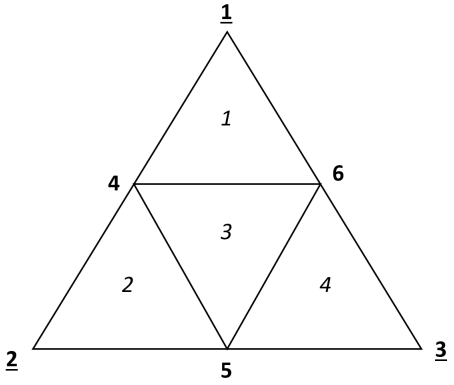


Figure 12: Gmsh element formed by four BEMLIB elements

In figure 12, bold underlined numbers indicate point indexes shared by BEMLIB and Gmsh. Numbers in italic refer to the four BEMLIB elements present within one Gmsh element defined by the largest triangle.

The Gmsh triangular elements, defined by 3 points, are split. Thus, the midpoints of the corners of each triangle are calculated, stored and added to the end of the connectivity matrix.

Finally, to apply the boundary conditions at the wing surface, the normal vectors pointing toward the fluid at each point must be calculated. These vectors are computed by the routine that reads the mesh created by Gmsh, although they must be flipped at some points to ensure they point toward the fluid. This verification can

[2]Here the connectivity matrix refers to an array that describes which points belong to a given element.

be done by setting a point inside the wing (e.g. the average location of the points $\vec{O_w}$), from that point, since the wing is completely convex, the scalar product of the vector pointing toward each surface point ($\vec{P}$) and the normal vector at that point must be positive ($\vec{O_wP} \cdot \vec{n} > 0$), otherwise, the normal vector is inverted. This algorithm is explained in more detail the integration point mask section in figure 16.

Once the variables are correctly computed, they are saved in a file that LNM_3D is able to read. When this is completed, Matlab executes LNM_3D which in turn generates the solution of the superficial potential and writes it in an output file.

DNS data reading

In order to compute the volume integral term in the force decomposition (see equatio (12)), one needs flow data in the control volume selected. In this work, this flow data is read from a DNS performed in the CFD group of the Bioengineering and Aerospace Engineering department of the Universidad Carlos III de Madrid.

This DNS is solved using the immersed boundary method applied to the full Navier-Stokes equations. This method involves two meshes isolated from each other, there is a mesh defining an object in the flow (i.e. a wing) and a mesh defining the control volume. Since both meshes are constructed independently, the points on both of them, do not necessarily coincide with each other and there are volume points inside the immersed object. Since, in principle, there are no pints on the control volume mesh on which the boundary conditions can be directly set, smooth constraints are imposed at the boundary surface. This leads to smeared solutions where the flow variables are smeared at the body wall. The surface mesh used in this analysis is one very similar to the wing shown in figure 11 and for the volume mesh, a structured staggered mesh such as the one in figure 13 is used.

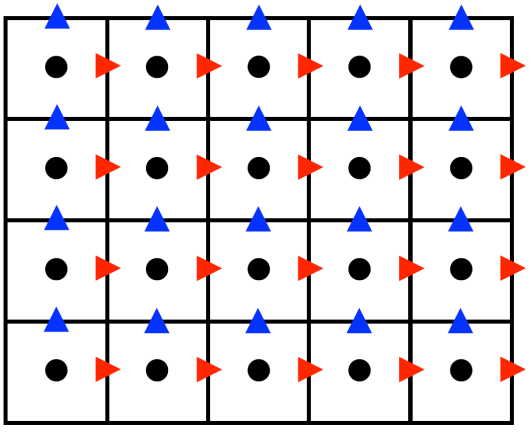


Figure 13: Example of a 2D staggered grid

In figure 13, the red and blue triangles represent x and y velocities respectively and the black dots are the pressure points or cell centers.

With the velocity field, the vorticity is computed using second order spatial finite differences. This computational approximation of the derivative yields that at the corners of each cell. Note that, the velocity, the vorticity and the pressure fields are computed at different locations within each cell. This cannot be used directly and requires the interpolation of their values on the same point within each cell. This was done using linear interpolation, i.e. for the vorticity, the 4 values at the corners of a cell were averaged onto the center and for the velocity, the values at opposing faces, were averaged to obtain the value at the cell center.

When all the fields at the pressure points are calculated, one requires only the gradient of the potential at these same points to compute the integral. The calculation of these gradients is discussed below.

Control volume potential gradient calculation

To complete the set of variables required for the volume integral in the force decomposition, one requires the gradient of the potential. This in principle can be done in two different ways, one can calculate the potential itself and then compute the gradient by finite differences or compute the gradient directly. Each approach has advantages and disadvantages which are explained in this section.

Having the some potential on the surface of the wing it is possible to calculate ϕ at each cell center by integrating Green's third identity over the wing using the following equation:

$$f(\vec{x_0}) = \frac{1}{4\pi}\int_S [-\frac{1}{r}\nabla f(\vec{x}) + f(\vec{x})\frac{\vec{x_0}-\vec{x}}{r^3}] \cdot \vec{n}(\vec{x})\ dA(\vec{x}), \tag{15}$$

where $\vec{x}$ represents the position vector of a point, the subindex 0 denotes points in the control volume, the domain S is the surface of the wing and the function f is the potential and $r = |\vec{x_0} - \vec{x}|$.

Once ϕ is calculated at the center of each cell, the gradient can be computed by means of second-order spatial differences as the vorticity in the DNS data reading part of section 2.5.

On the other hand, the gradient can also be calculated through:

$$\frac{\partial f(\vec{x_0})}{\partial x_{0i}} = \frac{1}{4\pi}\int_S \frac{x_i}{r^3}[\vec{n}(\vec{x})\nabla f(\vec{x})]dS(\vec{x}) + \sum_{j=1}^{3}\frac{1}{4\pi}\int_S f(\vec{x})\Big(\frac{\delta_{ij}}{r^3} - 3\frac{x_i x_j}{r^5}\Big)\vec{n}_j(\vec{x})dS(\vec{x}), \tag{16}$$

where the subindexes i and j denote the 3 Cartesian directions. δ_{ij} is Kronecker's delta.

Obviously, calculating the gradients directly with equation (16) is more accurate than taking finite differences. There is an errors associated to the finite difference method. Moreover, the gradient is computed at the midpoint between the cell centers and must be interpolated back to the pressure points adding another error source.

Calculating ϕ in the CV using equation (15) is 1.5 to 2 times faster than equation (16) (Taking into account the finite differences and interpolation required for equation (15)). This is very important since a high density of points is used in the volume mesh. Since the differences in gradients obtained with both methods are larger than 5%, it was decided to calculate them using the more accurate equation (16).

This part of the main Matlab routine is the one that takes the longest due to the size of the control volumes used. To alleviate this, much attention was paid to optimizing the operations, vectorizing Matlab computations wherever possible.

Integration point mask

Due to the embedded boundary method used for the DNS, there are pressure points that lay within the volume defined by the wing. These points are not part of the volume integral and must be removed before it is computed.

Moreover, the values of the flow variables as well as the potential gradient are not to be trusted close to the wing surface. In the case of the flow variables (velocity and vorticity), the embedded boundary method used yields smeared results at points near the boundary.

In figure 14, one can see a zoomed in view of a x-y cut of the whole control volume at the midspan of the wing. The vorticity in the z direction, ω_z, is represented here at the mid downstroke. Thus showing the leading edge vortex developing as well as some trailing vortices. However, at the wing boundary, there is a smooth transition where a sharp change (a wall) should be.

This makes the solution at the points close to the wall invalid to be used in the force decomposition algorithm.

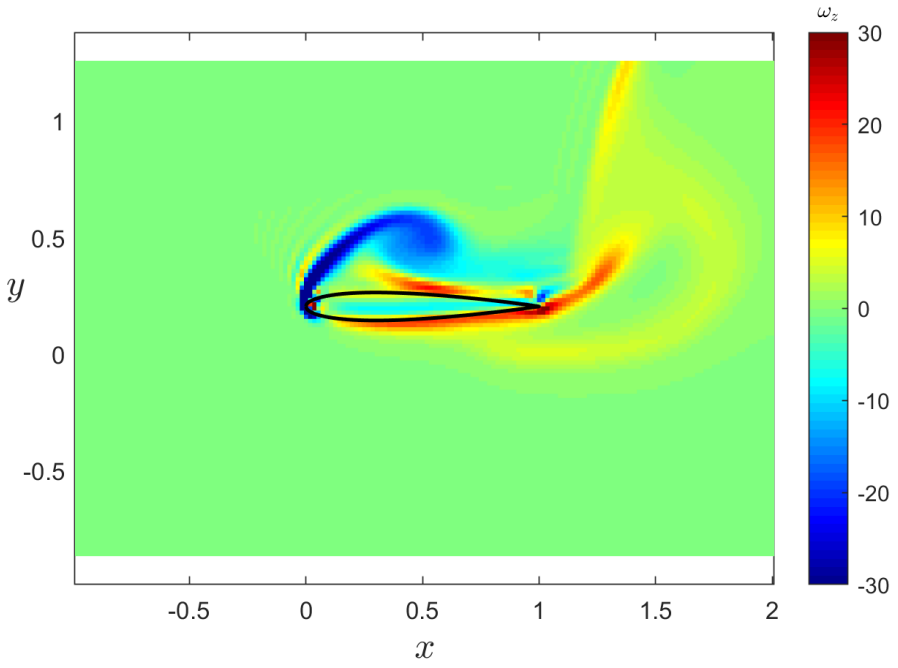


Figure 14: Vorticity in the z direction

The problems associated with the gradient computation, come from the terms defining the integrand in equation (16). In this equation, there are terms that go with the inverse of the radius to the fifth power. This means that if a point in the surface mesh and the volume mesh are too close to each other, the value of the gradient yielded is huge and since the number of points in both meshes is finite, unevenness in the gradients close to the surface is expected.

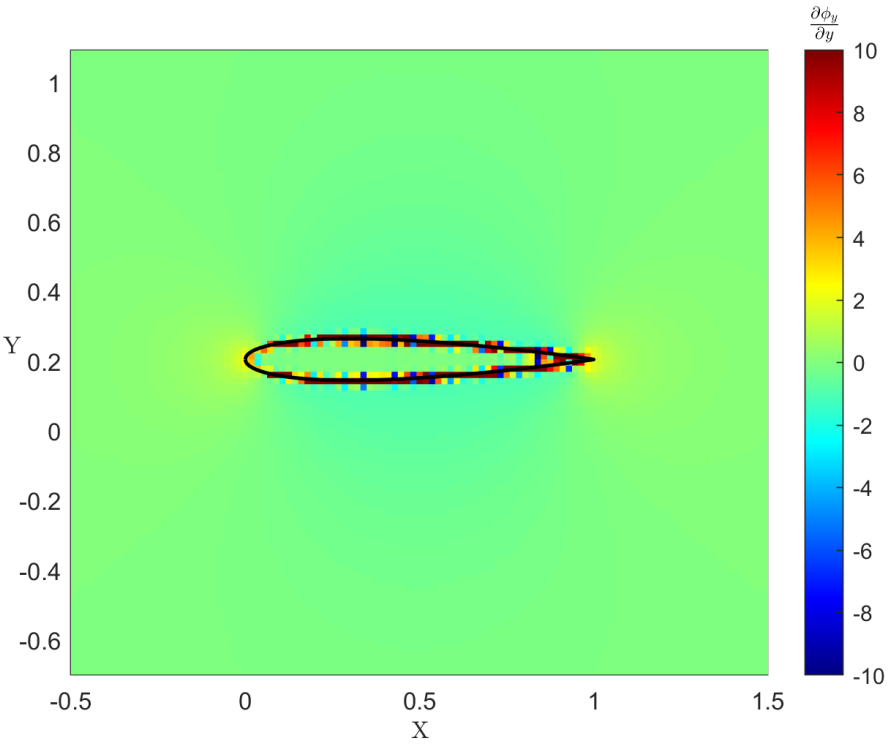


Figure 15: Gradient of ϕ_y in the y direction.

In figure 15, one can see that the gradient is smooth all through the volume except for the points very close to the wing, where it is substantially different.

To circumvent these problems, the mask is extended to the points close to the wing as well as the points within the wing.

To determine which cell centers are inside the wing, a routine was created that finds the closest point in the surface mesh to each pressure point in the volume mesh.

A sketch of the algorithm used to determine if a cell is inside or outside the wing is shown in figure 16.

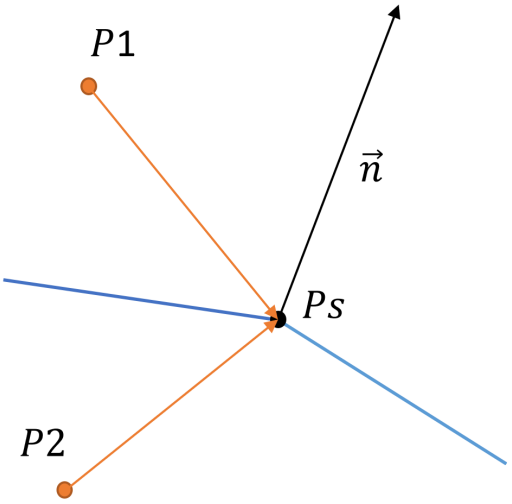


Figure 16: Sketch of points near the surface.

In figure 16, a typical case in the determination of which points are inside of the wing. The dot product of the vector $P1\vec{P}s$ and $\vec{n}$, is negative thus it must be outside the wing. On the contrary, the dot product of $P2\vec{P}s$ and $\vec{n}$ is positive forcing it to be inside the wing.

This algorithm works as long as the curvature on the surface is not too large in comparison with the density of surface points. If this happens, as is shown in figure 17, even though the point is outside the wing, the angle between $P1\vec{P}s$ and $\vec{n}$ is larger than 90^{o}. Thus it acts as if it was inside the wing.

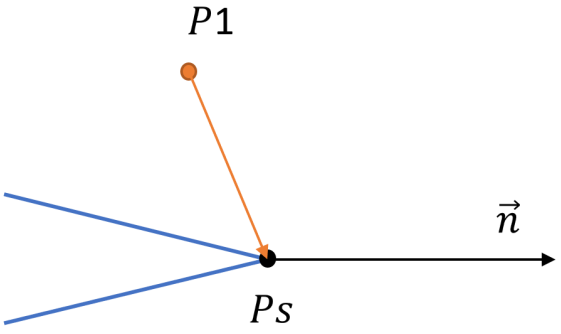


Figure 17: Sketch of the effects of curvature on the algorithm.

This causes minor problems near the trailing edge of the wing , where a sharp edge is found. Causing some negligible errors later on in the force decomposition.

Then, to filter out all points close the wing as well, a routine was used that calculates the distance from each point to the surface of the mesh. From that one can identify all points to close to the wing. The objective here, is to remove at least one layer of points around the wing the minimum number of layers necessary.

For efficiency, not all points are checked to be inside or outside the wing. In order to avoid unnecessary computations, a bounding box around the wing can be defined. All points outside the box are known to be one. The algorithm to build the mask then, is only applied to those points inside the bounding box. Saving 80% to 90% of the time it would take without the use of the bounding box.

Finally, this mask is nothing else than a matrix the size of the control volume, with a one if the corresponding point is outside the wing, or a zero if the point is inside or very close to the boundary. Figure 18, shows a cut made by a plane in the xy direction of the mask at the midspan. Yellow represents a one (outside the wing) and blue represents a zero (inside the wing).

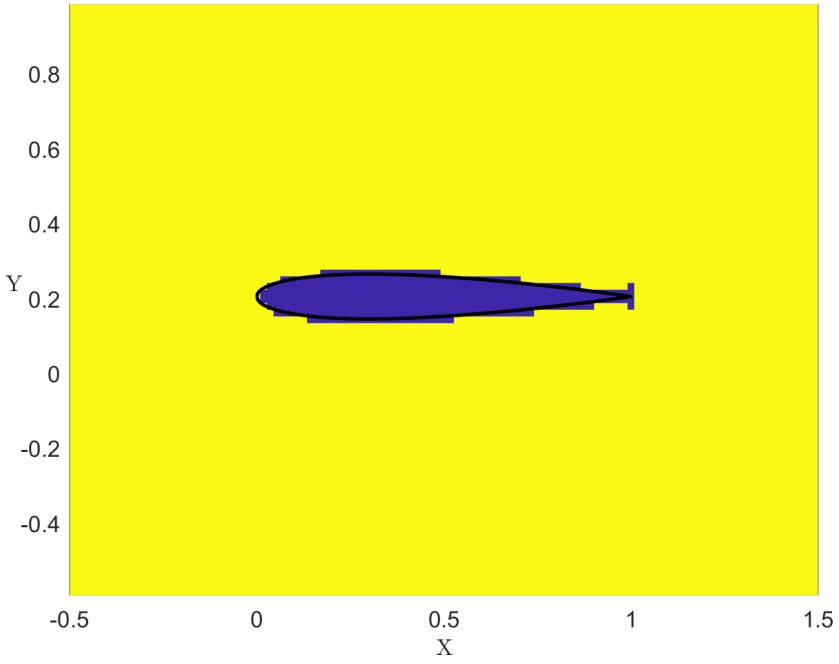


Figure 18: Resultant mask around the wing.

In figure 18, an xy cut of the wing at the midspan is shown to illustrate a typical mask used to solve the spurius solution of the gradients near the wing surface. The mask includes points up to 1% of the chord perpendicular to the wing surface. Because of the 56 points per chord definition of the volume mesh, this includes up to one point outside the wing. Typically, in this project, up to two points where included in the mask. Notice that 'up to' one point is said, since the mesh is completely structured, with 1% distance specified, one or no point is taken. If 2% is used, two or one points are taken from the boundary.

Force integration

Finally, each contribution of the force is computed with the variables obtained in the other parts of the routine following equation (12).

For the added mass terms on (12), an integration routine was built to calculate the surface integral. The integration assumes a constant value of the function over each BEMLIB element[3] and equal to the average of the values at the corners. This is where the frontal mesh algorithm from Gmsh used becomes useful. Were the elements not very close to equilateral, barycentric interpolation must have been used for each element.

The vorticity within the flow term is a volume integral. However, since the volume mesh is completely structured and uniform, volume elements are always hexahedrons of constant size. Thus the integral is just the sum of the value of the integrand at each point multiplied by the volume of one volume element.

Finally, the total force is obtained from the DNS solution and the equation is solved to compute the surface vorticity term.

2.5.2 Mesh convergence

The fineness of the mesh is critical in any case of numerical analysis, it determines the accuracy as well as the time cost of the analysis. It is therefore crucial to optimize this parameter.

In the case studied, it was decided that the best way to do this was to calculate the surface potential in several stages of mesh refinement and then look at the maximum difference in the potential on the surface between each case and the most refined one analyzed. The relative error of the potential flows computed for this convergence are shown in figure 19.

[3]BEMLIB elements are those formed by splitting each element of the Gmsh mesh into 4 as shown on figure 12.

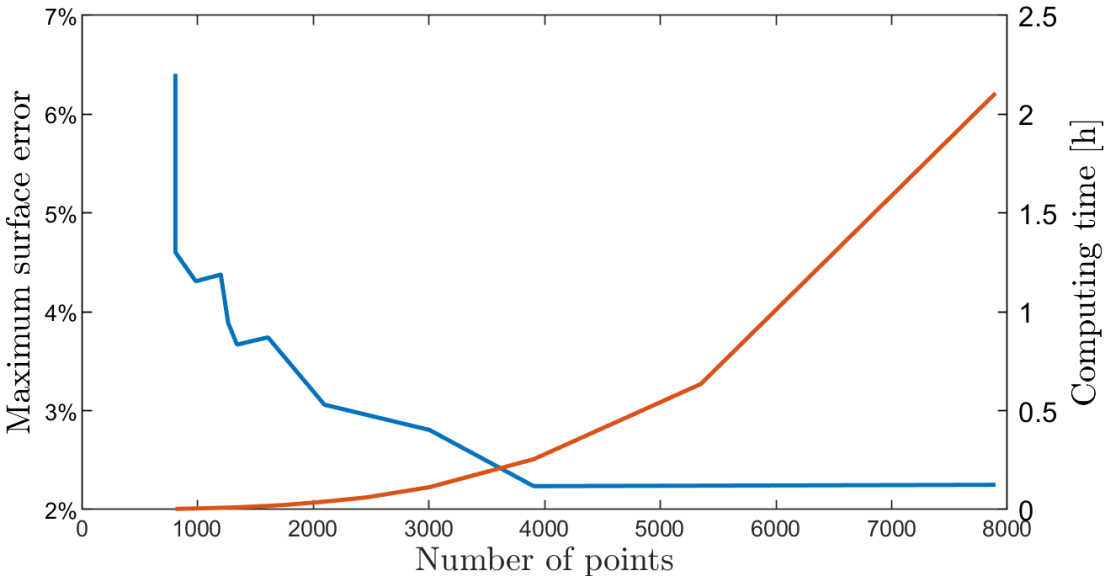


Figure 19: Maximum error on the surface potential with respect to the most refined case and time to compute potential.

Figure 19 shows how from 4000 point to the largest resolution, the error is maintained at $\approx 2.3\%$. It is important to note that to compare the results of this potentials between the 2 meshes, the low fineness mesh results were interpolated on to the most refined case, this results in some errors inherent to the interpolation that may be causing the small resultant error. Because of this, any mesh with $>\approx 4000$ points can be considered valid.

3 Results

Having an understanding of how the program works, the focus can now shift to present the outputs of the algorithm and discuss their behavior.

Case description

Due to a lack of time, only one case was studied as a proof of concept, although the method in principle applies in general to any type of motion of any wing.

The studied wing has a rectangular planform, no geometric twist and $AR = 2$ and unitary chord (c). It is built with NACA 0012 airfoils.

The movement described by the wing is a heaving sinusoidal motion without rotation. The motion is defined as:

$$y = h_z \cos(kt) \tag{17}$$

Where the amplitude $h_z = c$ and the reduced frequency $k = \frac{2\pi fc}{U_\infty} = 1$, being f the heaving frequency of the motion. The incoming flow is governed by a Reynolds number $Re = 500$.

The details of this case and its computational setup can be found in [6].

In this work, we have used 32 frames which contain the flow data during the whole cycle. Only at these frames, will the force decomposition be computed.

As was said before, 2 meshes make up the problem, one mesh defines the control volume in which the wing is flying and the other defines the wing itself. The volume mesh is completely structured and defined by a point density of 56 points per chord. The surface mesh is triangular, unstructured and has a larger number of points than 4000 as specified necessary in the mesh convergence section of the paper.

Finally, there are two parameters that must be adjusted manually to fine tune the performance of the method. The first one is the distance from the surface of the wing that will be included in the mask (taken out of the integration) and the other is how much of the control volume will be taken into consideration.

This last one is particularly critical, for one, it greatly influences computing time (together with the surface mesh refinement) so it must be as small as possible. On the other hand, it must be as big as necessary to include all the relevant flow characteristics.

Before, a snapshot of the vorticity was shown (figure 14). In it one can see vortices spanning the full control volume. However, from equation (16), one can see that the gradient goes to 0 with the inverse of the distance cubed. To observe this, a density of thrust and lift can be defined as:

$$\delta_t = \frac{2(\vec{u} \times \vec{\omega}) \cdot \nabla \phi_x}{U_\infty^3 cb} \tag{18}$$

$$\delta_l = \frac{-2(\vec{u} \times \vec{\omega}) \cdot \nabla \phi_y}{U_\infty^3 cb} \tag{19}$$

This is the adimensionalized integrand of the vorticity within the flow term (third term in equation (12)). Plotting this in a cut of the control volume along the midspan yields:

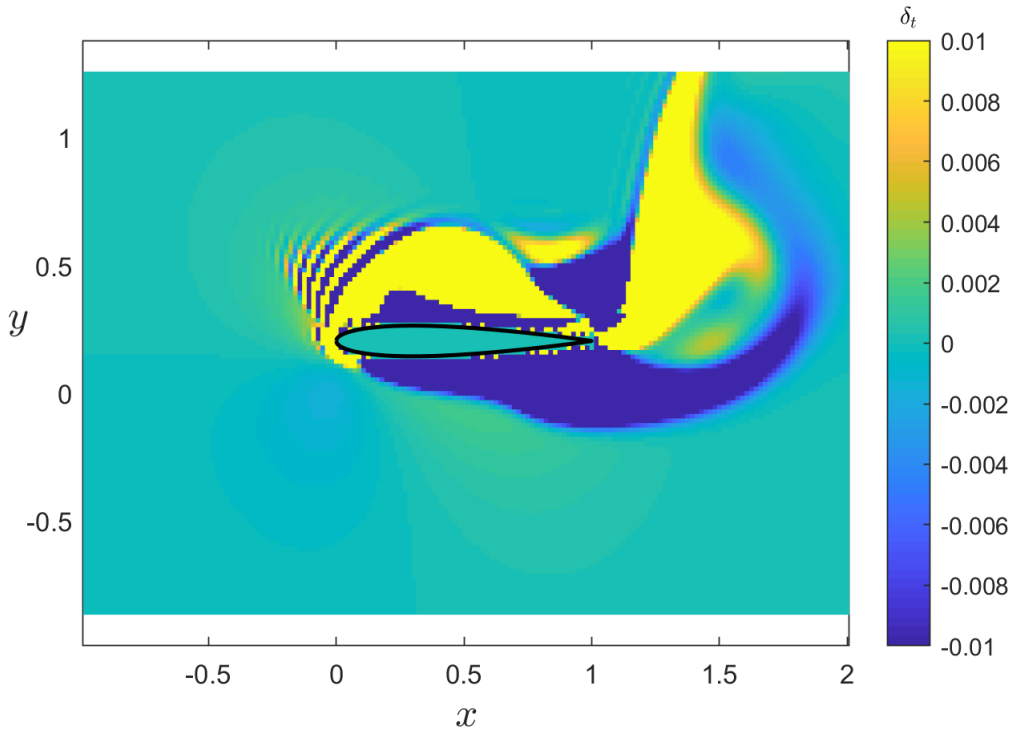


Figure 20: Density of thrust.

From figure 20, one can see that the brunt of the vorticity within the flow integrand is very close to the airfoil with minor non-zero locations being outside the box defined in this case which takes one chord in each direction from the wing.

Similar results are obtained for δ_l which is shown in figure 21 for completeness:

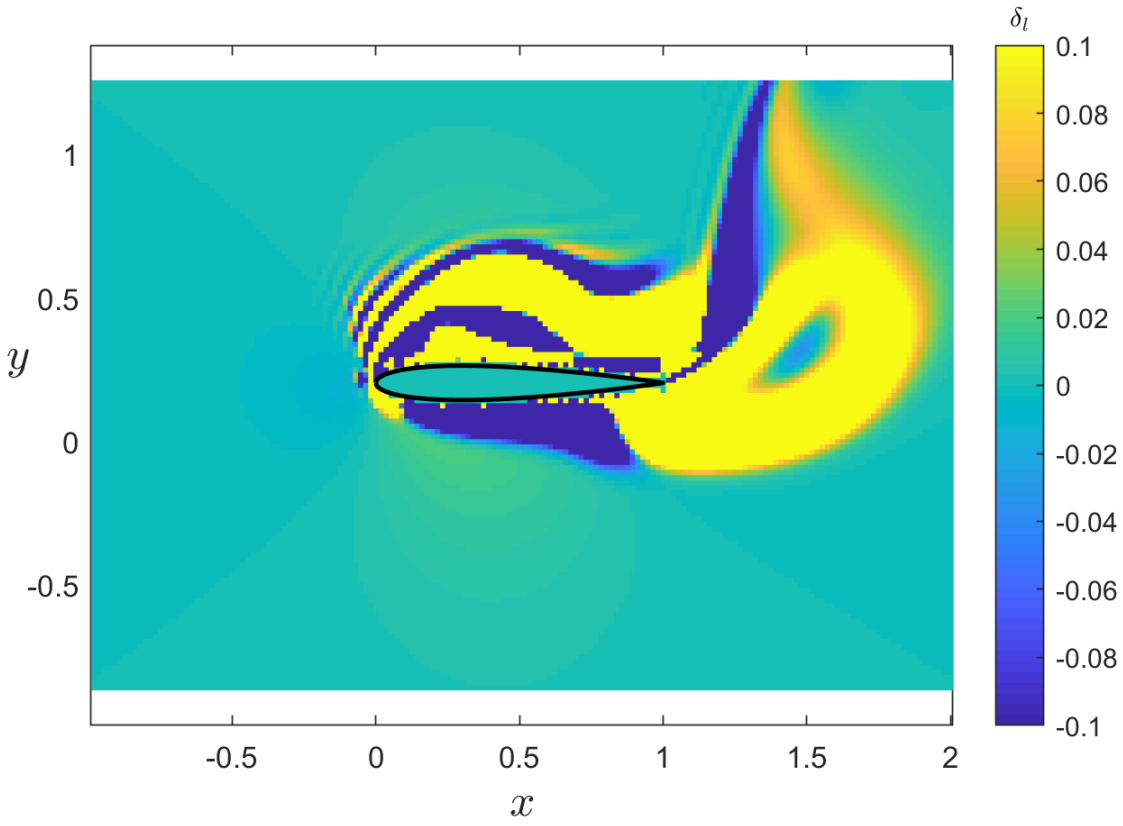


Figure 21: Density of lift.

Knowing this, and because of the prohibitive time taken to use a larger control volume, it was decided to use for the final results a CV consisting on one chord from the wing.

3.1 Force decomposition of a heaving wing

In this section, the results of the case described in the previous one are presented and discussed. Moreover, they will be compared to a DNS solution from [6] in order to certify the validity of the program. Also, the problems affecting its accuracy will be described.

The analysis will be broken down into the forces in the y and x direction starting with the former.

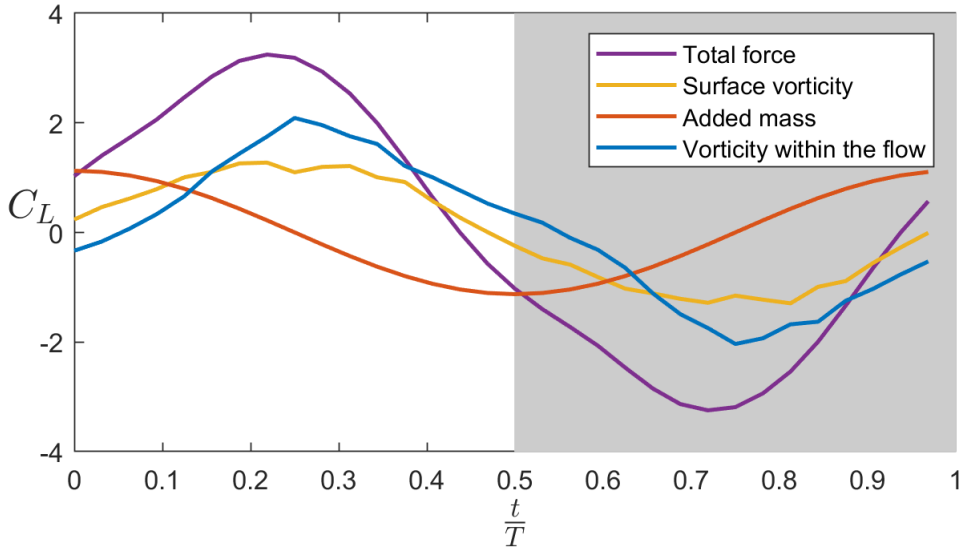


Figure 22: Force decomposition in the y direction.

Figure 22, shows the lift decomposition during one period of motion. Recall that the wing starts at its highest point and then drops, reaching its minimum at $t = 0.5T$ to begin its ascent again. White background represents the wing moving down and gray background the up-stroke motion. It makes sense then, looking at the total force, that as the wing goes down, lift appears, and the opposite happens on the way up.

The total force, obtained from a DNS analysis, is the sum of the other three lines on the plot: The surface vorticity, the vorticity within the flow and the added mass.

The added mass term, as is known, is the direct response of the fluid to the motion of the wing. For instance, as the wing starts to accelerate at the beginning of the movement, the fluid around it remains at rest effectively resisting the acceleration. The same happens at the middle of the motion, the fluid is moving down with the

wing and as deceleration happens, the fluid slams into the wing causing a negative force.

In the lift direction, there is a large contribution from this term because the area perpendicular to the direction of travel (up-down), is very large.

Returning to figure 22, the next contribution to be discussed will be the vorticity within the flow term.

In flapping flights, the detached vortices produced influence greatly the forces acting on the wing, recall that, this term is calculated as:

$$F_V = -\rho \int_V \frac{\nabla \phi_i}{U_\infty} \cdot (\vec{u} \times \vec{\omega}) \, dV \tag{20}$$

So not only the vorticity influences the force, but the gradient of the potential in the volume as well. This potential decreases with the inverse of the distance to the wing to the fifth power, so very close vortices are much more important in this contribution.

The main vortex in flapping flight is the leading edge vortex, which trails behind the wing as it moves and detaches from the leading edge. This is shown in figure 23 at the bottom of the motion.

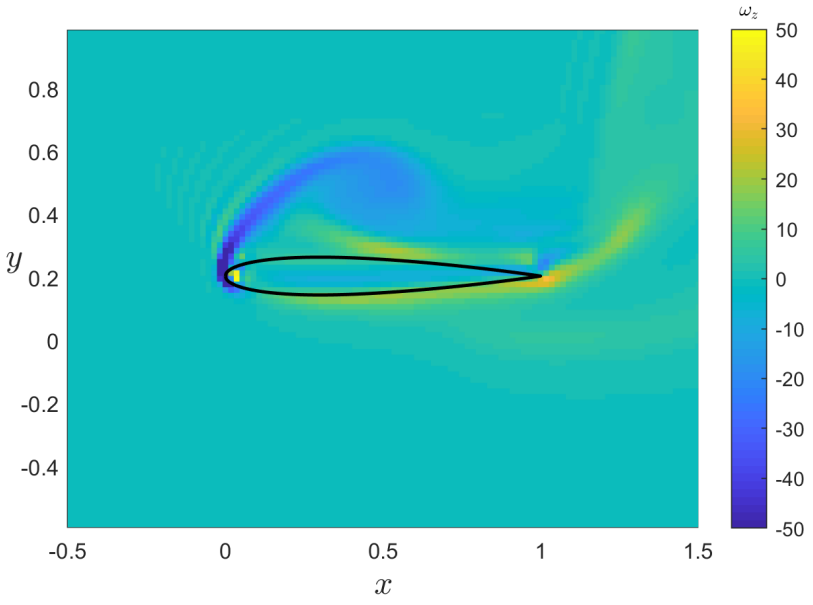


Figure 23: Vorticity in z showing the leading edge vortex.

Although not shown here, it is intuitive that the leading edge vortex is largest when the wing moves at highest velocity down or upwards. For this reason, this contribution is largest at $t = 0.25T$ and $t = 0.75T$.

Finally, the last contribution to the force is the surface vorticity. This term, is produced by the forces acting tangentially to the surface of the wing in the chordwise (x) direction.

As was said before, this term is not computed directly but rather it is calculated from the difference of the other terms against the total force.

Now the focus shifts to the forces along the x direction, i.e., the drag or thrust.

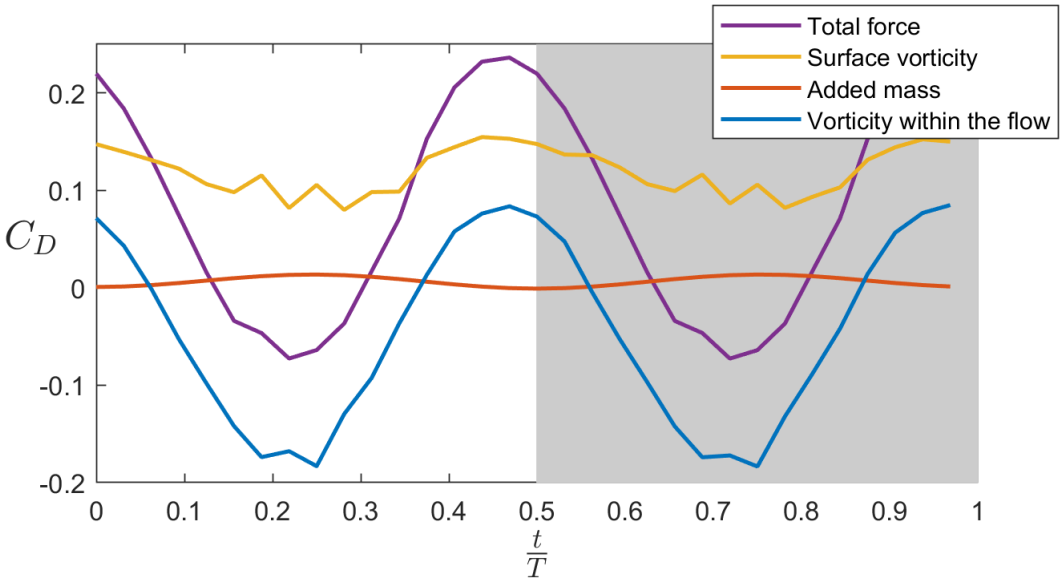


Figure 24: Force decomposition in the x direction.

Although in the y direction, each contribution was of the same order of magnitude, in figure 24, one can see that this is not the case in the x direction.

The added mass term, which contributed to the lift force significantly, it is however almost 0 in the x direction force. This is a result of the direction of the motion, since the wing is moving up and down, there is barely any fluid dragged along the x direction.

The main term in the total force is the vorticity within the flow. As was said above, vortices generate a large amount of the force in flapping flight. Here, the maximum forces are produced at the maximum velocity and acceleration points, with a slight lag related to the time taken for vortices to develop.

Finally, the surface vorticity term, is almost as large as the vorticity within the flow. It gives the biggest contribution to the drag, balancing the thrust given by the vorticity within the flow term.

3.2 Comparison of the results to a validated case

Having presented the results obtained from the case, it is important to discuss the accuracy of the results comparing what was obtained with known results to reflect on the problems of the program in its final version.

To measure the accuracy of the program one can use the surface vorticity term. Recall that the surface vorticity is the only term that is not calculated directly but is computed as the difference between the added mass and vorticity within the flow term and the total force. Therefore, it can be used to check the accuracy of the model in cases such as this one where the flow was previously solved and the viscous force term is already known.

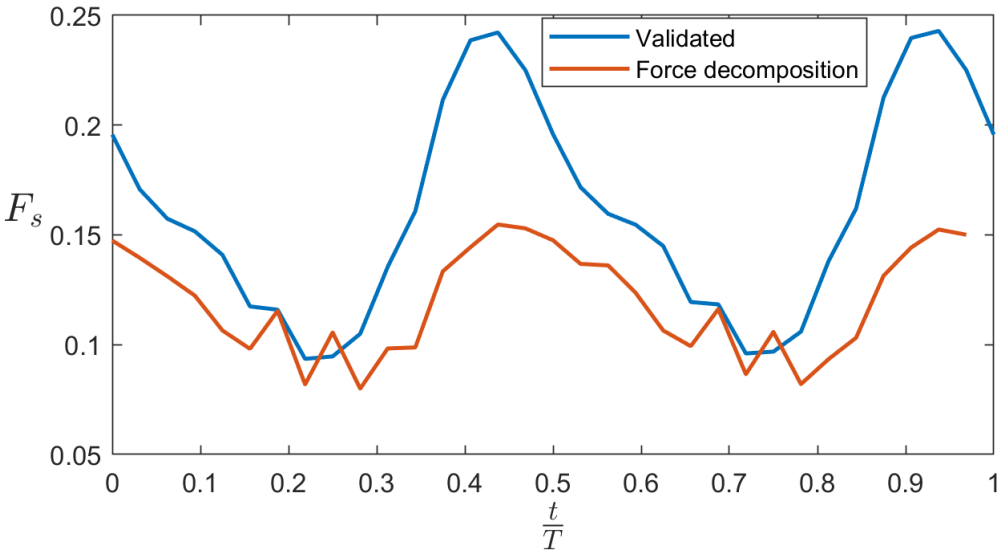


Figure 25: Surface vorticity term in the x direction comparison.

Looking at both lines in figure 25, one can see that the surface vorticity term has the same features as the validated force. The peaks happen at the same time, slightly behind the maximum acceleration points ($t = \frac{n}{2}T$ for any real n) and the valleys coincide as well. However, the peaks in the validated solution are significantly higher and there is a jaggedness in the valleys in the force decomposition term.

All of the differences in figure 25 can be explained by the volume mesh definition employed. Recall that, one of the main parts of the decomposition program was building a mask that takes the points inside of the wing out of the integral. This mask also needed to be expanded past the surface of the wing to neglect singularities in the potential gradient and the spurious flow variables at the wing interface.

In general, to do this one must take at least the first point from the surface away. However, with the volume mesh definition being only 56 points per chord, this means that the minimum distance necessary from the wing surface is 2% of the chord.

Recall that the vorticity within the flow contains the potential gradient in its formulation and that, in turn, decreases with the inverse of the distance to the wing to the fifth power (12). Moreover, large magnitudes of vorticity are found at a region close to the wing, where the boundary layer develops. Combining these effects, results in larger contributions to the vorticity within the flow term contained in the closest points to the wing.

If one needs to remove at least 2% of the chord of this first region, the solution will change drastically.

Because of the neglected points, the peaks do not reach as high value as they should. Similarly, as was said, 2% chord removal is the minimum necessary. With that distance most of the singularities are avoided but not all of them. For that reason, at certain points through one period, jaggedness appears where some of these singularities were not filtered by the mask.

An effort to increase the point density was made by multiplying the point density by 2 (112 points per chord) by interpolating the flow variables onto the new mesh. Although promising, the interpolation yielded too many errors producing an erratic behavior as the distance to the wall of the neglected points decreased. Because of it, this approach was abandoned.

4 Conclusions

Within the context of this project, a program was developed to obtain the contributions to the force described in the decomposition algorithm proposed by Chang [2]. The decomposition requires a surface potential flow to be calculated. To that end, the library BEMLIB, was modified to solve the required potential flow and was validated against previous known results.

During the development of said program, several problems were encountered related mainly with the numerical nature of the operations performed, such as integrals and interpolations.

The first problem encountered was the computational requirements to calculate the gradient of the potential flow in the control volume. This computation, requires a surface integral over the surface of the wing performed for each point of the control volume. For large sizes of CV, this is a calculation that requires a high amount of computing resources. While this problem was not completely solved, the computations were optimized to the point where the problem is minimized and the algorithm can run in a reasonable time while using a sufficiently large control volume.

The second problem relates to the inaccuracy of the solutions close to the wing surface. The errors in this region come from the immersed boundary method used in the DNS to compute the flow variables and from the calculation of the gradient of the potential. If one point in the wing surface is too close to a point in the control volume, the potential gradient at that point in the volume is very large because the gradient at any point in the volume is proportional to the inverse of the distance from each surface point to the fifth power. To minimize the effects of this problem, the program neglects the points too close to the wing in the integration of the vorticity within the flow term.

Neglecting the points close to the wing however, neglects a large part of the integrand of the vorticity within the flow term.

The solution to this problem then is to increase the density of the volume mesh. If this mesh is more dense, then the strip of neglected points close to the wing can be thinner and the resultant contributions more accurate.

Due to a lack of time, this could not be done in this work. However, using the current mesh density, the results obtained in the force decomposition are similar in shape and in order of magnitude to those in the DNS solution from [6]. Proving that the tool works correctly and can provide very accurate results if only the density of the volume mesh is increased.

5 Future work

With the program working, it is important to discuss the following work that could build on top of it. This section is devoted to explaining possible work that could be spurred by the tools and results discussed in this work.

First of all, as was said in the conclusions, the program needs to be tested with denser meshes. The program requires an accuracy upgrade if it is going to be used with any success.

To do that, the same case must be computed with DNS to obtain the flow variables in the new mesh. The main problem then, will be computational time. Not only one needs to perform the DNS analysis but the decomposition algorithm will then take significantly longer to finish. This will entail further optimization in the code, both in computing performance, to allow faster results and in memory efficiency, to allow the use of larger meshes. With all this one could theoretically increase the accuracy to acceptable levels.

If the program is finally completely validated, the next logical step is to apply the decomposition algorithm to a full library of cases, not only a single wing in heaving but two wings in flapping flight as well. With this library, an analysis of the contribution of each term in the decomposition could be performed and obtain meaningful relationships between the motion of the wing or wings and each term of the forces. Advancing the understanding in the characteristics of flapping flight and unsteady aerodynamics.

These relationships will then allow for the creation of simpler models that relate the geometry and motion characteristics of the wing and the forces acting on it.

The models will allow less powerful machines to predict forces acting on flapping wings. This may allow their introduction in the control systems of drones based on flapping flight.

6 Budget

In this section, the theoretical costs of the project are discussed. This includes costs related with the salary of the researcher, the costs of tools required and the costs of the hardware used for computations.

The average cost per hour of a junior engineer in Spain is 10€. The total research time invested is 360 hours. That makes 3600€ the cost from the salary of the researcher.

Most software tools used are free or free for students, such as Gmsh and the Intel Fortran compiler used to compile BEMLIB to run efficiently. However, Matlab is not free for students and the license for it adds an additional 500€ to the total costs of the project.

Finally, for the most time consuming operations, a node in a cluster of the Bioengineering and Aerospace Engineering department was used. Assuming an average cost of 0.2€ per core and computing hour, and a total amount of 1000 computing hours using a 12 core machine. The costs for the hardware rises up to 2400€.

Summing up, if all costs are added, the cost of the whole project rises up to 6500€ approximately.

7 Socio-economic impact of the project

Flapping flight has shown enormous potential since the very early studies performed on the subject. Some studies even show that in could be more efficient than fixed wing traditional flight[4]. Nevertheless, flapping wings are not practical for commercial flight because the inertia of the required wing would be too large. Making it difficult to move and impossibly heavy due to the structural requirements.

However, there is a sector where flapping flight could revolutionize current technology, the micro vehicle and drone market. Current micro flyers rely on multiple small propellers. Its efficiency would be massively improved if flapping wing technology were to be introduced. Moreover, in this sector where every gram on the vehicle counts, there is quite a high constraint in the size of the batteries used in flight. A more efficient propulsion and lift system would yield smaller vehicles to accomplish current missions and higher range possibilities.

In addition to that, the size of this vehicles means that they will be fragile. So they must be able to avoid obstacles with quick reactions. In slow flight, where these drones perform at its best, flapping wings massively outperform fixed wings in maneuverability. For reference on this, one can look at the reactions of flies [4].

The current drone market is expected to be valued at $100 billion in 2020 in an estimation by Golman-Sachs in 2016[5], most of the value being in the military market with $\approx$ $70 billion. In this industry, flapping drones could carry out from search and rescue to intelligence gathering missions much more efficiently than current technology and with increased capabilities than with current technology.

FMAVs are currently being designed and built without a full understanding of the design principles required for efficiency in such flight. As was said in the first section of this work, its objectives are to try to breakdown the resultant forces to simpler components that can be then analyzed individually to gain more insight into the aerodynamic characteristics of the problem. If finally the problem is cracked and really efficient flapping vehicles start to appear, the market will be quickly flooded with them for their increased capabilities and simpler construction.

[4] More information here.
[5] Goldman-Sachs report.

8 Regulatory framework

MAVs are the most likely area affected by flapping flight technology. FMAvs are regulated under the same framework as typical propeller drones. In Spain, the current regulations for remotely controlled aircraft are described in 'Real Decreto 1036/2017'[6]. This recent change in the regulation comes from the need to expand the previous law set following the current rate of expansion of the recreational drone market.

For recreational use, this regulation mainly limits the use of drones in populated areas and the times at which flight is permitted. For professional use, there are additional licenses required to control a drone, which are not necessary in recreational use.

However, flapping flight is a new technology that is expected to change substantially the current market when it is put into production. Thus, more regulatory changes are expected before it reaches widespread use.

[6] BOE 29/12/2017

References

[1] M. A. Ashraf, Young J., and J. C. S. Lai. Reynolds number, thickness and camber effects on flapping airfoil propulsion. *J. Fluids Struct.*, 27(2):145–160, 2011.

[2] C.-C. Chang. Potential flow and forces for incompressible viscous flow. In *Proc. R. Soc. Lond. A*, volume 437, pages 517–525, 1992.

[3] C. P. Ellington. The aerodynamics of hovering insect flight. IV. Aerodynamic mechanisms. *Philos. Trans. R. Soc. Lond., B, Biol. Sci.*, 305(1122):79–113, 1984.

[4] S. N. Fry, R. Sayaman, and M. H. Dickinson. The aerodynamics of free-flight maneuvers in drosophila. *Science*.

[5] C. Geuzaine and J.-F. Remacle. Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. *Int. J. Numer. Methods Eng.*, 79(11):1309–1331, 2009.

[6] A. Gonzalo, G. Arranz, M. Moriche, M. García-Villalba, and O. Flores. From flapping to heaving: a numerical study of wings in forward flight. 2018.

[7] G. C. Lewin and H. Haj-Hariri. Modelling thrust generation of a two-dimensional heaving airfoil in a viscous flow. *J. Fluid Mech.*, 492:339–362, 2003.

[8] A. Martín-Alcántara, R. Fernandez-Feria, and E. Sanmiguel-Rojas. Vortex flow structures and interactions for the optimum thrust efficiency of a heaving airfoil at different mean angles of attack. *Phys. Fluids*, 27(7):073602, 2015.

[9] M. Moriche. A numerical study on the aerodynamic forces and the wake stability of flapping flight at low reynolds number. 2017.

[10] C. Pozrikidis. *A Practical Guide to Boundary Element Methods with the Software Library BEMLIB*. CRC Press, 2002.

[11] W.H. Press, Teukolsky S.A., Vetterling W.T., and B.P. Flannery. *Numerical Recipes in Fortran 77: The Art of Scientific Computing*. Cambridge University Press, 1992.

[12] L. Quartapelle and M Napolitano. Force and moment in incompressible flows. *AIAA J.*, 21(6):911–913, 1983.

[13] W. Shyy, H. Aono, C. K. Kang, and H. Liu. *An introduction to flapping wing aerodynamics*. Cambridge University Press, 2013.

[14] S. Wang, X. Zhang, G. He, and T. Liu. Evaluation of lift formulas applied to low-reynolds-number unsteady flows. *AIAA J.*, 53(1):161–175, 2014.

[15] Z. J. Wang. Vortex shedding and frequency selection in flapping flight. *J. Fluid Mech.*, 410:323–341, 2000.